Paper ML08

Sparc AI: A Native Metadata-Driven Engine for End-to-End Clinical Trial Analysis

Anil Mogha, K3-Innovations Inc., New Jersey, USA
Ashwin Kumar Nyalakonda, K3-Innovations Inc., Hyderabad, India
Gourav Garg, K3-Innovations Inc., Hyderabad, India

ABSTRACT

Purely metadata-driven automation in clinical trials provides strong regulatory alignment and deterministic outputs but struggles to adapt to evolving protocols and complex study-specific nuances. Conversely, general-purpose Large Language Models (LLMs) offer flexibility and rapid development but lack native CDISC awareness, traceability, and compliance required for regulatory submissions. This paper presents **SPARC AI**, a hybrid, metadata-native clinical trial automation platform that combines deterministic metadata-based engines with LLM-augmented intelligence.

SPARC operates directly on CDISC standards such as SDTM, ADaM, and Define.xml, enabling automated ADaM specification creation, dataset generation, and TLF production with full traceability. LLMs are used selectively for contextual code generation, metadata enrichment, validation, and documentation—always grounded in structured metadata and deterministic outputs. Pilot implementations demonstrate significant reductions in programming timelines and audit-ready outputs suitable for regulatory submissions. An additional benefit of SPARC's unified platform is the seamless handoff it enables between biostatistics, programming, and medical writing teams, reducing workflow friction while preserving end-to-end traceability.

This paper outlines the architecture, design principles, and practical deployment strategies of hybrid AI in clinical data engineering, illustrating why metadata-centric, LLM-augmented automation represents a sustainable future for regulatory-compliant clinical trial workflows.

INTRODUCTION

Clinical trial data analysis workflows are undergoing rapid transformation driven by increasing study complexity, compressed timelines, and heightened regulatory scrutiny. Organizations are under pressure to deliver high-quality, submission-ready outputs faster while maintaining strict compliance with CDISC standards and regulatory expectations.

Historically, clinical programming workflows have relied on a combination of manual specifications, SAS or R programs, document-centric TLF shells, and fragmented tooling. While these approaches are regulator-accepted, they are labor-intensive, error-prone, and difficult to scale.

Recent advances in automation and AI have introduced new possibilities. However, current solutions tend to fall into two extremes:

- Metadata-based deterministic automation, which is compliant and reproducible but rigid
- General-purpose LLM-driven approaches, which are flexible but opaque and difficult to validate

This paper argues that neither approach alone is sufficient. Instead, a **hybrid architecture**—where metadata remains the system of record and LLMs operate in constrained, auditable roles—offers a practical path forward.

EVOLUTION OF AUTOMATION IN CLINICAL TRIAL WORKFLOWS

Clinical trial automation has progressed through phases: script-centric reuse, metadata-driven standardization, and AI-assisted augmentation. Each phase improves speed, but introduces trade-offs in transparency, maintainability, and compliance. SPARC AI is designed to combine the strengths of deterministic automation (control, traceability, reproducibility) with the strengths of LLMs (semantic understanding, adaptation, code scaffolding), while reducing the risks of either approach used in isolation.

LIMITATIONS OF PURE METADATA-BASED DETERMINISTIC AUTOMATION (MDA)

Metadata-based deterministic automation (MDA) provides a strong structural foundation for clinical trial analysis workflows by enforcing consistency, traceability, and regulatory alignment. In the context of ADaM and TLF generation, MDA ensures that dataset structures, variable definitions, and standard derivations are applied consistently across studies. However, while MDA excels at rule execution, it lacks the adaptive "intelligence" required to manage the nuanced, study-specific logic inherent in statistical analysis.

Static Mapping Logic

MDA is highly effective for simple, one-to-one mappings (e.g., copying variables from SDTM to ADaM). However, many ADaM derivations involve **complex, context-dependent logic**, such as baseline definitions, windowing rules, censoring logic, or imputation methods (e.g., last observation carried forward). These derivations vary across studies based on protocol design, SAP interpretation, or regulatory feedback. Encoding all possible variations into a universal deterministic rule library is impractical and leads to brittle systems requiring manual intervention.

Mock Shell Fragility in TLF Development

Statistical mock shells are typically authored manually in Word or Excel. These artifacts capture layout, grouping, and intent rather than executable logic. MDA systems cannot inherently interpret these visual templates or infer analytical intent from them. As a result, shells must be manually translated into code, revisions require repeated manual updates, and traceability between shell, program, and output is fragmented.

Validation Bottlenecks

Traditional validation approaches rely on double programming, which is duplicative and slow. MDA can validate whether outputs conform to predefined rules, but cannot evaluate whether the **contextual meaning** of a TLF aligns with SAP intent. For example, MDA can confirm that counts and percentages were computed correctly, but cannot assess whether the population, endpoint definition, and grouping logic reflect the intended analysis described in the SAP.

LIMITATIONS OF PURE LLM-BASED APPROACHES

LLMs introduce flexibility and speed, but pose risks in regulated workflows: non-deterministic outputs, weak traceability, and lack of native CDISC awareness. Without deterministic control layers, LLM-generated programs are difficult to justify during audits or regulatory questions. These issues make pure LLM approaches unsuitable as primary engines in clinical trial submission pipelines.

HOW AI AND LLMs BRIDGE THE GAPS

AI—specifically LLMs—complements deterministic automation by interpreting **semantic intent**. When used within a controlled, metadata-native framework, AI can address key limitations of MDA without becoming the system of record.

Context-Aware Code Generation

LLMs can ingest structured metadata alongside natural language inputs such as ADaM specifications, SAP sections, and TLF descriptions. Using this context, LLMs can generate SAS or R code for complex derivations, study-specific transformations, and TLF programs aligned with the analytical intent. In SPARC, LLM-generated code is not free-form; it is grounded in metadata context and validated by deterministic layers before execution.

Intelligent Interpretation of Mock Shells

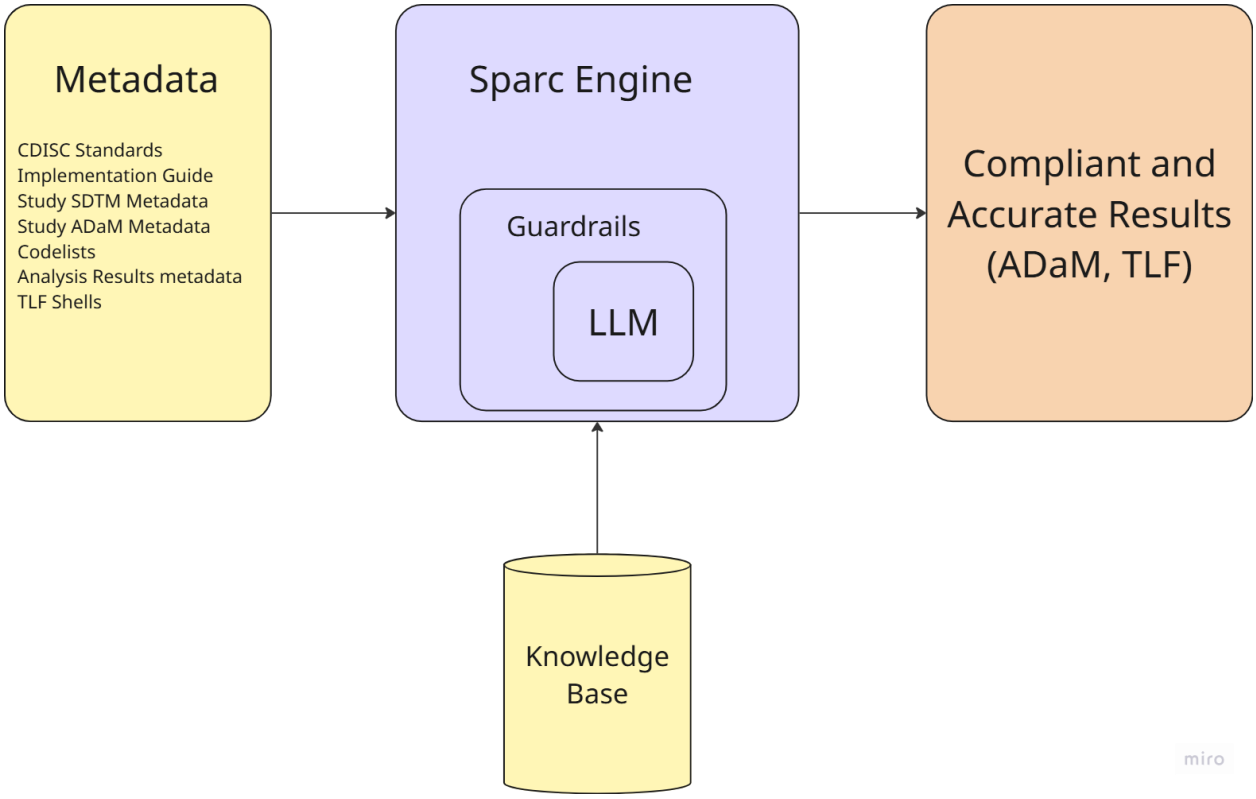
AI can interpret mock shells as expressions of analysis intent rather than static templates. By analyzing titles, footnotes, grouping structures, and descriptive text, AI can infer required datasets, analysis populations, grouping and sorting logic, and summary methods—reducing manual translation.

Semantic and "Visual" Validation of Outputs

Beyond rule-based checks, AI enables an additional layer of validation by "reading" final outputs similar to a human reviewer. This includes verifying whether a produced table aligns with the underlying ADaM dataset and whether the output reflects SAP-defined populations and groupings. This approach complements deterministic checks and reduces reliance on full double programming.

SPARC AI ARCHITECTURE OVERVIEW

SPARC is designed as a metadata-native platform for regulatory workflows:



- **Metadata-native core**: operates on CDISC metadata and standardized representations (e.g., SDTM/ADaM structures, Define.xml concepts)
- **Deterministic control layer**: ensures reproducibility, versioning, and auditability
- **LLM augmentation layer**: supports enrichment, scaffolding, semantic validation, and study-specific flexibility, constrained by metadata context and validation gates

A CONCRETE EXAMPLE OF HYBRID CODE GENERATION FOR TLFs

A practical way to operationalize hybrid automation is to generate code in **bounded, labeled blocks**, where deterministic code is produced from metadata and LLM-generated code is limited to specific "surface areas" that benefit from semantic reasoning. The following example illustrates SPARC-style hybrid generation for a demographics and baseline characteristics table (Table 14.1.3.1) using R packages commonly used in clinical reporting.

In this pattern:

- **Metadata-generated blocks** provide the stable scaffolding: libraries, dataset loading, column templates, title/export conventions.
- **LLM blocks** fill in the study-specific logic: preprocessing, factor level handling, variable analysis list, and label alignment.

- The **LLM is constrained** because the structure and expected “slots” are already fixed by deterministic automation, reducing the chance of misinterpretation and limiting where errors can occur.

Hybrid-generated R code example (metadata blocks + LLM block)

```
# ============================================================================
# TLF: Table 14.1.3.1
# Generated: 2026-01-15 17:05:05
# Study: Xanomeline-100-Spandex
# ============================================================================

# ===== LIBRARY_BLOCK_START =====
# Generated: 2026-01-15 17:05:05
# Library loading code - Generated based on metadata
library(tern)
library(rtables)
library(dplyr)
library(haven)
# ===== LIBRARY_BLOCK_END =====

# ===== DATASET_BLOCK_START =====
# Generated: 2026-01-15 17:05:05
# Loading dataset from source
library(haven)
adsl <- read_sas("AdamDatasets/Outputs/adsl.sas7bdat")
# ===== DATASET_BLOCK_END =====


# ===== COLUMN_TEMPLATE_BLOCK_START =====
# Generated: 2026-01-15 17:05:05
# ColumnTemplate Name: Treatment Columns
column_template <- basic_table() %>%
split_cols_by("TRT01A", split_fun = keep_split_levels(c("Placebo", "Xanomeline Low Dose", "Xanomeline
High Dose"))) %>%
add_overall_col("Total") %>%
add_colcounts()

# ===== COLUMN_TEMPLATE_BLOCK_END =====

# ===== LLM_BLOCK_START =====
# Generated: 2026-01-15 17:05:05

# Preprocessing
adsl <- adsl %>%
mutate(
AGE = as.numeric(AGE),
AGEGR1 = factor(AGEGR1, levels = c("<65", "65-80", ">80")),
SEX = factor(SEX, levels = c("F", "M")),
RACE = factor(RACE),
WEIGHTBL = as.numeric(WEIGHTBL),
HEIGHTBL = as.numeric(HEIGHTBL)
) %>%
filter(!is.na(TRT01A) & TRT01A != "")

# Analysis Implementation
lyt <- column_template %>%
analyze_vars(
vars = c("AGE", "AGEGR1", "SEX", "RACE", "WEIGHTBL", "HEIGHTBL"),
var_labels = c(
"Age (y)", "Age Group", "Sex", "Race",
```

```
"Baseline Weight", "Baseline Height"
)
)

# Build Table
tbl <- build_table(lyt, adsl)
# ===== LLM_BLOCK_END =====

# ===== TITLE_BLOCK_START =====
# Generated: 2026-01-15 17:05:05

# Main title
main_title(tbl) <- "Table 14.1.3.1"

# Subtitles
subtitles(tbl) <- c("Summary of Demographic and Baseline Characteristics")

# Print table
print(tbl)
# ===== TITLE_BLOCK_END =====

# ===== EXPORT_BLOCK_START =====
# Generated: 2026-01-15 17:05:05
# Export table output

export_as_rtf(tbl, 'TLFs/Outputs/t_table_14_1_3_1.rtf', landscape=FALSE)
# ===== EXPORT_BLOCK_END =====

# ============================================================================
# Code generation completed for TLF: Summary of Demographic and Baseline
Characteristics
# ============================================================================
```

Why this reduces LLM error surface area

This code demonstrates a core principle of SPARC's hybrid approach: **use deterministic automation to establish the contract**, and use LLMs only within explicitly bounded regions.

In this example, deterministic automation:

- Fixes the library requirements and version expectations
- Fixes data loading conventions and locations
- Fixes the column structure and treatment-arm ordering
- Fixes titling and export format conventions

The LLM block is constrained to:

- Type normalization and factor level handling
- Selection of analysis variables and readable labels
- Minimal data filtering logic consistent with the template

Because the structure and "slots" are already present, the LLM has less room to hallucinate or rewrite core components. This pattern also improves auditability: reviewers can see exactly which regions are metadata-driven versus AI-assisted.

PILOT STUDY OUTCOMES

In pilot studies, SPARC demonstrated:

- 85–95% reduction in ADaM specification and TLF generation timelines
- Improved consistency and reduced manual rework during SAP amendments
- Generation of audit-ready outputs with end-to-end traceability

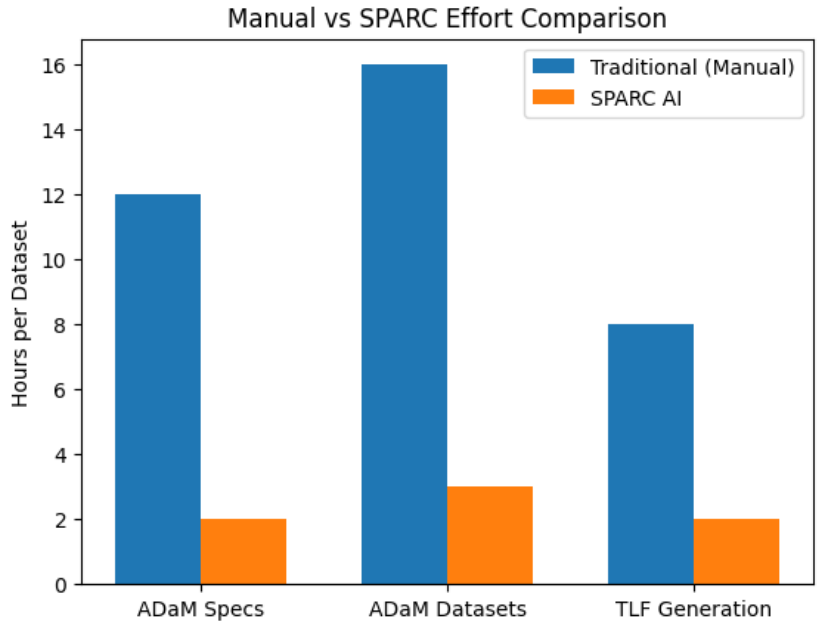


Figure 1. Reduction in Programming Effort per Dataset Using Hybrid AI Native Metadata-Driven Automation

These outcomes reflect both time savings and reduced operational risk in late-stage trial activities.

REGULATORY AND COMPLIANCE CONSIDERATIONS

SPARC's hybrid design is built for regulated environments:

- Metadata-driven determinism as the system of record
- Versioned audit trails and reproducible regeneration
- Clear separation of AI-assisted content from authoritative logic
- Validation gates ensuring AI outputs remain consistent with metadata and SAP intent

This supports transparency and defensibility during internal audit and external regulatory review.

CONCLUSION

Pure deterministic automation provides compliance and consistency but lacks the flexibility required for nuanced statistical workflows. Pure LLM approaches provide adaptability but lack the traceability, reproducibility, and CDISC alignment required for submissions. SPARC AI demonstrates a practical hybrid model: deterministic, metadata-native scaffolding paired with constrained AI augmentation for study-specific logic, enrichment, and semantic validation.

This hybrid pattern reduces the surface area where LLMs can introduce errors, while unlocking measurable efficiency gains. Metadata-centric, LLM-augmented automation offers a sustainable path toward faster trials, lower operational burden, and improved submission readiness—without compromising transparency and compliance.

REFERENCES

1. https://sparcgen.com/
2. GPT-4o Model | OpenAI API

3. cdisc-org/sdtm-adam-pilot-project
4. Create Common TLGs Used in Clinical Trials • tern
5. insightsengineering/rtables: Reporting tables with R
6.

ACKNOWLEDGMENTS

The authors thank **Kajal Anandani**, CEO of K3-Innovations, for her support and guidance. We also thank **Daniela Popa** and the **K3-Innovations team** for their collaboration and feedback during the development and pilot implementation of SPARC AI.

RECOMMENDED READING

- CDISC ADaM Implementation Guide
- PHUSE White Papers on Automation and AI
- Regulatory guidance on computer system validation

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Author Name: Anil Mogha
Company: K3-Innovations, Inc.
Email: anil.mogha@k3-innovations.com
Website: https://k3-innovations.com

Author Name: Ashwin Kumar
Company: K3-Innovations, Inc.
Email: ashwin.kumar@k3-innovations.com
Website: https://k3-innovations.com

Author Name: Gourav Garg
Company: K3-Innovations, Inc.
Email: gourav.garg@k3-innovations.com
Website: https://k3-innovations.com